REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 06-10-2009	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 23-Jan-2008 - 22-Jan-2009

4. TITLE AND SUBTITLE Global Sensor Management: Military Asset Allocation	5a. CONTRACT NUMBER W911NF-08-1-0051
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 667100
6. AUTHORS Thom J. Hodgson	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES North Carolina State University Office of Contract and Grants Leazar Hall Lower Level- MC Raleigh, NC 27695 -7214	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO 11. SPONSOR/MONITOR'S REPORT NUMBER(S) 54311-MA.1

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
The USAF maintains a network of sensors for a variety of purposes including detecting threats, collecting intelligence, and monitoring space. Because of its nature, the network must be able to complete all its varying missions with consistently high probability. Thus, there is a need to assign the sensors to tasks and functions so as to maximize the network's capability to meet its objectives. While it is possible to determine the best allocation based on a total enumeration of potential sensor assignments, this is intractable for large problems. Once an

15. SUBJECT TERMS
surviellance, sensors

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Thom Hodgson
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR		19b. TELEPHONE NUMBER 919-515-5194

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Global Sensor Management: Military Asset Allocation

Johnathon L. Dulin[**]
Kristin Arney[+]
Thom J. Hodgson[*]
Ben J. Lobo[*]
Curtis M. Mears[#]
Reha Uzsoy[*]

[*]Edward P. Fitts Department of Industrial & Systems Engineering
North Carolina State University
Raleigh, NC 27695

[**]Concurrent Technologies Corp.
Johnstown, PA 15904

[+]Department of Mathematics
United States Military Academy
West Point, NY 10996

[#]U.S. Army Special Operations Command
Fort Bragg, NC 28310

Abstract

The United States maintains a network of sensor assets for a variety of purposes including detecting threats, collecting intelligence, and monitoring space. Because of its nature, the network must be able to successfully complete all its varying missions with consistently high probability. Thus there is a need to assign these sensors to tasks and functions so as to maximize the network's capability to meet its objectives. While it is possible to determine the best allocation based on a total enumeration of potential sensor assignments, this is intractable for large problems. Once an allocation scheme is determined, some sensors may need reassignment in response to certain types of events, or sensors may simply fail. We address these issues in developing a heuristic to find optimal/near optimal solutions to sensor networks. Computational experiments demonstrate the ability of the heuristic to achieve high-quality sensor allocations within the time constraints inherent in the dynamic environment in which it operates.

This research was supported by the United States Strategic Command (USSTRATCOM) through the Army Research Office (ARO)

Introduction: The United States military maintains an integrated air defense system that may be characterized as a sensor network, dedicated to the accomplishment of several tasks, including missile defense, missile warning, space surveillance and strategic and tactical intelligence collection. Each task involves a series of functions that must be performed in order to successfully accomplish the task. This network can be represented by a set of nodes and arcs, where the nodes represent the performance of a function by a particular sensor and the arcs represent the capability of one sensor to communicate information to another sensor.

The ultimate objective of planners is to assign sensors across the four tasks above to achieve high probability of successfully accomplishing each of the tasks. Because resources are limited, the allocation of sensors to tasks presents a challenge to planners. Assigning a sensor to one task may prevent it from being assigned to another. The network itself is well defined. There is a clear definition of which sensors can perform which functions and which sensors can communicate with one another. Each sensor has a unique set of functions that it can perform with differing probabilities of success, depending on the sensor's own limitations and the current state in which it is working. Using this information to develop the 'best' allocation scheme is the subject of this paper. The challenge here is two-fold. The first is to determine an appropriate initial allocation of sensors to tasks and the ability to reassign the sensors as time and circumstances dictate; the second, to accomplish this in near real-time.

This problem is of interest to the United States Strategic Command (USSTRATCOM). Their mission is to "provide the nation with global deterrence capabilities and synchronized Department of Defense (DoD) effects to combat adversary weapons of mass destruction worldwide. Enable decisive global kinetic and non-kinetic combat effects through the application and advocacy of integrated intelligence, surveillance and reconnaissance; space and global strike operations; information operations; integrated missile defense and robust command and control" (Joint Publication 3-14, 2002). Among these national objectives, missile defense, missile warning, space surveillance and intelligence collection are of greatest concern (Joint Publication 3-01, 2007). In this paper, we first define the problem of allocating sensors to tasks, and then develop a method for rapidly obtaining near-optimal solutions.

The remainder of the paper is organized as follows. We first present a definition of the problem and of the network. We then review previous related literature, including the broad areas of network characteristics and analysis and resources allocation solution approaches. Finally, we present our solution approach and the computational experiments necessary to test its efficacy, concluding with key findings and suggestions for future work.

Problem Description: There are four *tasks* that form the basis of the problem: missile defense, missile warning, space surveillance, and strategic and tactical intelligence collection. Each task has unique characteristics and requirements, and also shares common characteristics and requirements with others. In particular, for all tasks the flow of information between sensors follows the same general processes. Thus, a general network representation can be applied across each task.

Missile defense is a critical task whose failure yields an imminent threat to the United States. The objective of this task is to detect the occurrence of a (possibly) unpredicted missile event (e.g., a ballistic missile launch), track its path, classify its level of threat and, if necessary, engage and eliminate it. Space surveillance and strategic and tactical intelligence collection are also important, but for different reasons. Each is a form of information gathering. Space surveillance monitors the movement of man-made objects in space, while scientific and technical collection is concerned with gathering information from various locations on earth. In each case, the overarching goal is essentially the successful collection of information.

A *function* is a specific type of activity that a sensor must complete successfully in order to contribute to the successful accomplishment of an assigned task. The functions are governed by precedence constraints, with the preceding function needing to be completed before the subsequent function can commence. The functions considered here, in order of their precedence order, are as follows:

1. Monitor: Each sensor conducts surveillance over a specified region of earth or space, or is placed in standby mode for later use.
2. Detect: An event occurs when one or more sensors are monitoring a region, and that event is noted by the sensor(s).
3. Cue: A detected object's path is projected through space and time based on its current position and other available data, such as trajectory.
4. Track: An object of interest is monitored with the express intent of determining its trajectory.
5. Classify: Based on an object's trajectory and other available data, a determination is made as to whether or not the object is a threat.

6. Update: Position, trajectory and threat classification of the object are recorded continuously and adjusted, which may require repetition of previously activated functions.
7. Engage: Depending on the classification of an object, resources are assigned to either avoid or eliminate the threat.
8. Kill: An object classified as a threat is engaged and destroyed.
9. Assess: When an event that requires military action occurs, such as the engagement of enemy troops or the launch of a missile defense projectile, the result of that action is evaluated.

The *tasks* and *functions* form the basis for the representation of the problem as a sensor network. In order to illustrate the proposed representation, it is best to describe a scenario representative of the environment in which the model will be used. This scenario depicts a notional network for missile defense (Figure 1) similar to one that might be encountered in the employment of the model. It encompasses the characteristics of the actual environment and is valid for illustrative purposes only. The functions previously defined are grouped vertically. Some of the sensors represented in the model (e.g. phased-array radars) have sufficient capacity to perform their functions in support of multiple events and across multiple tasks. However, ***assignable*** sensors may only be assigned to a single task at a time, although they may perform multiple functions within that task. For example, the Monitor function can be performed by any combination of sensors 1 through 6. These six shaded sensors are defined as *assignable* sensors, and are allocated to only one of the four tasks at a particular time. These are the sensors of interest to this paper. The un-shaded sensors are *shared*, and can perform their functions in support of all tasks simultaneously. Thus they are not assigned to a particular task, but are available to all tasks.

Each node in the network represents a sensor performing a specified function. For example, the node labeled as “Sensor 7” performs function 2. The arcs represent the transfer of information from sensors performing a particular function to those performing the subsequent function. For example, the arc between Sensor 10 (performing function 5) and Sensor 12 (performing function 6) indicates that Sensor 12 can receive the data needed to perform its function from Sensor 10. The other incoming arc indicates that Sensor 12 may also receive the necessary data from Sensor 9. The shading indicates which assignable sensors may be allocated to which tasks: Sensors 1 and 2 may be assigned to either Task 1 or Task 2; Sensors 3 and 5 may be assigned to either Task 1 or Task 3; and Sensors 4 and 6 may be assigned to either Task 1 or Task 4.

Given this description, the problem can be stated as follows: Allocate the assignable sensors to tasks in order to maximize the probability of successfully completing Task 1, subject to maintaining the probability of success for the remaining tasks above stated thresholds. A decision support tool addressing this problem must be flexible enough to deal with continuous changes to the environment in which it is operating.

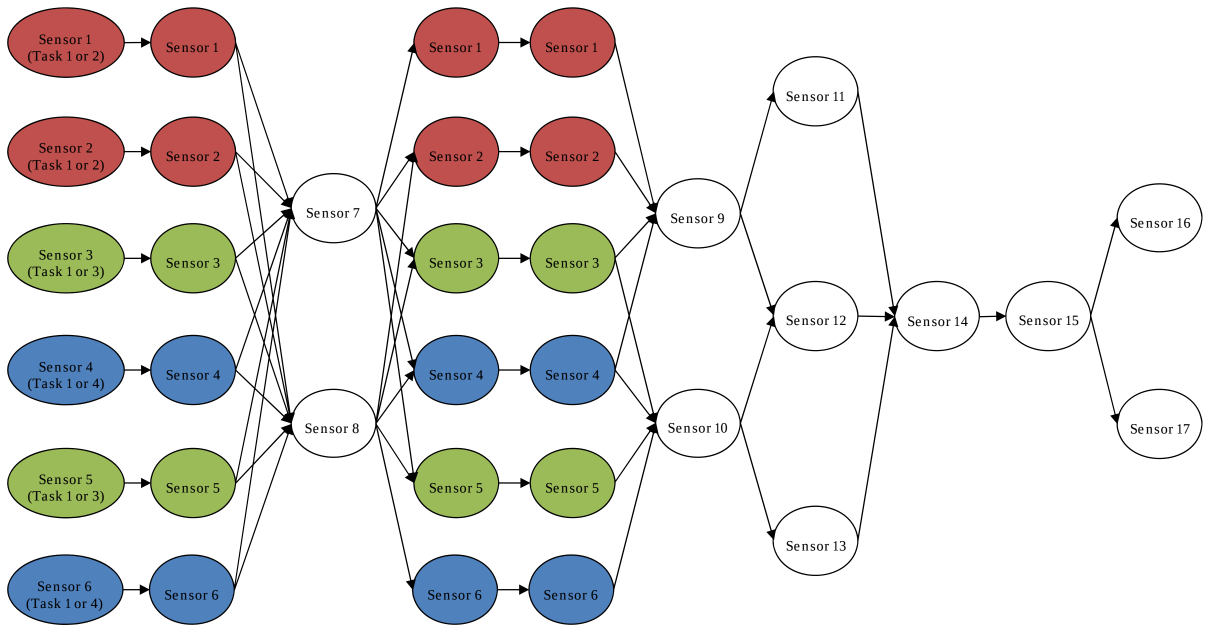

Figure 1: Notional Network, Task 1

Assumptions: As with any modeling effort, a number of assumptions are necessary to define the problem and to help shape the approach used to obtain a solution. Generally, assignable sensors may be capable of performing more than one function; but they may only be assigned to a single task at a time. It is assumed that if a sensor performs any function while assigned to a particular task, it will remain assigned to that task until it performs all of the functions related to that task that it is capable of. Referring to Figure 1, if Sensor 1 is assigned to Task 1, then it will perform functions 1, 3 and 4 in support of Task 1. This reduces the need for excessive communication between sensors, minimizing potential loss of information and time lag. It therefore stands to reason that a sensor that has successfully performed one function is better able to perform the subsequent function (providing it has the ability to do so) than another sensor that performed a different function previously. So the reassignment of a sensor from one task to another before it performs all its functions related to the first task likely will not yield an assignment with higher overall probability of task success than an assignment that allows it to complete all assigned functions. For these reasons we examine only single assignment allocations. Shared sensors are assumed to be able to perform their functions across multiple tasks, and they are implicitly allocated to all tasks at all times. It is also assumed that no sensor is left unassigned, as cost is not a factor.

In order to efficiently estimate the probability that a given sensor allocation will be able to perform the tasks required of the network, we must assume a limited form of independence between sensors. Typically, in measuring network reliability, the assumption of independence indicates that the success of a sensor within the network does not rely on its prior history. In our case, however, the independence assumption refers to a specific relationship among sensors. For the set of sensors assigned a particular function of a given task in parallel, the probability that a particular sensor successfully completes that function is independent of whether or not another sensor successfully completes that function. However, the probability that a sensor successfully performs a particular function <u>is</u> dependent upon which sensor(s) successfully performed the preceding function and were able to communicate with the sensor in question. This due to the variance in accuracy of communicated data from the various sensors. As a result, while the basic tenets of reliability theory hold, significant modifications to the traditional network probability calculations must be made in order to accurately assess the performance of a network.

Analysis of historical information and test data provide adequate information to infuse the model with the probabilities needed to calculate the overall sensor, function and task success probabilities. Prior probabilities associated with the assignable sensors performing the first function are known, as well as conditional probabilities related to subsequent functions. Referring to Figure 1, the functions that must be performed to successfully complete the task are grouped vertically (i.e., sensors 1 and 2 both perform function 1). The known prior probabilities are the probabilities of success for Sensor 1 and Sensor 2. An example of a known conditional is Sensor 12 (performing function 6): the probability of success of Sensor 12 given Sensor 9 was successful ($P(12|9)$); and the probability of success given Sensor 10 was successful ($P(12|10)$). The probability of success for Sensor 12 given both Sensors 9 and 10 were successful is assumed to be the greater of the two individual probabilities: (i.e., $P(12|9,10) = max\ \{P(12|9), P(12|10)\}$). This assumption is not strictly true as the combination of information from two or more sources may increase the probability of success.

Finally, we assume that a network must have at least one feasible solution (allocation). A network solution is feasible if the minimum threshold probability of success is met for each of the three secondary tasks. Under this assumption the objective is to maximize the probability of success for Task 1, which is the most critical, subject to maintaining the success probabilities of the remaining tasks above a specified threshold.

Background: In developing an approach to the problem, there were several key areas that required investigation. To ensure that the sensor network is accurately modeled, significant effort was devoted to understanding the characteristics of sensors such as those employed by the Department of Defense. This included a review of sensor functions (Miranda, et al. 2007, Orman, et al. 1998), coverage (Howard, et al. 2002), prioritization and scheduling (Izquierdo-Fuente and Casar-Corredera, 1994; Pinedo, 2002), and control (Gordon-Spears and Kiriakidis, 2004).

By its very nature, this type of problem involves multiple uncertainties. For example, the detection of an event depends on the probability that a particular sensor can detect that type of event. Successfully tracking an object depends not only upon the probability that the sensor can track the object, but also on the probability that the previous event (object launch) was detected in the first place. No sensor is guaranteed to successfully accomplish every task it is assigned. The uncertainty can stem from failures due to conflicting signals or ineffective scheduling, such as an improper identification of an event (false positive) or a missed event (false negative). As more and more tasks are assigned to a sensor - be they multiple functions (surveillance, tracking, etc.) or multiple events (e.g. tracking multiple targets) - the capacity to successfully complete all functions or manage all events begins to diminish. This may necessitate a reallocation of resources to new functions or regions of surveillance, and typically affects overall performance of the network (Miranda, et al. 2007).

Resource allocation is an optimization problem that seeks to assign a set of resources to a set of activities to achieve the most desirable result - precisely the objective of the sensor network. In general, the problem can be formulated as:

$$\textit{Maximize: } \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

$$\begin{aligned} \textit{Subject to: } & \sum_{i=1}^{n} x_{ij} \geq 1,\ \forall\, j = 1,\dots,m \\ & \sum_{j=1}^{m} x_{ij} \leq 1,\ \forall\, i = 1,\dots,n \\ & x_{ij} \in \{0,1\}\ \forall\, i = 1,\dots,n,\ j = 1,\dots,m, \end{aligned} \tag{1}$$

where $x_{ij} = 1$, if resource *i* is assigned to activity *j*, and zero otherwise. In this case, resources *i* are assigned to activities *j* such that each activity is assigned at least one resource and each resource may be assigned to no more than one activity. The assignment of a resource to an activity yields a gain c_{ij} for the overall system, and the total gain is to be maximized (e.g. Wolsey, 1998). This gain may be any reasonable function by which the contribution of the assignment of a specific resource to the successful completion of a task may be measured. Often it is associated with revenue (or cost), but can be defined in other terms as well (e.g. system reliability). The common thread among most resource allocation methods is the focus on a single objective and the idea of a one-time solution. The secondary objectives that are addressed here add a layer of complexity, as does the dynamic element of a changing network environment. In the present case, computation of the objective function is a complex reliability calculation. A simple change of allocation may require significant computational effort. Thus, the use of a commercial optimization package is really not a viable option.

There is a need for real-time decision-making in the sensor allocation environment. What is required is a procedure that can take an existing sensor allocation and improve it in the face of changed circumstances, in a manner in which the procedure can be stopped at any time to yield a usable, if not necessarily optimal, solution. This is accomplished using local search, a family of heuristics in which an initial solution is progressively improved by local perturbations to the solution (referred to as moves). A similar approach has been suggested by Zweben et al. (1993), who use a local search base metaheuristic, specifically simulated annealing to develop a procedure for space shuttle ground processing. An extensive discussion of local search heuristics is given in Aarts and Lenstra (2003). While a number of results have been obtained on the performance of such procedures, these results are generally of the form of asymptotic convergence to a global optimum in a probabilistic sense. These are of limited applicability to the sensor allocation problem, since due to the time constraints in the application environment the likelihood of a solution procedure running long enough with a fixed set of data is very low.

Hence computational experiments (Barr, et al. 1995, Rardin and Uzsoy, 2001) are used to evaluate the performance of our proposed procedure.

Probabilities: Reliability theory gives us a methodology for evaluating how a group of sensors passes information from one end of a network to another (e.g. Kapur and Lamberson (1977). The reliability of a system that contains both serial and parallel characteristics can be said to represent the probability of successfully transmitting this information from end to end, is commonly known to be

$$\prod_{j=1}^{n}\left(1-\prod_{i\in P_i}\left(1-P(N_i)\right)\right)\prod_{i\in S}P(N_i), \tag{2}$$

where $P(N_i)$ represents the probability of success at node i, nodes in parallel are denoted by the sets P_i, and nodes in serial by the set S.

The specific problem at hand is to determine which allocation of assignable sensors produces the best probability of success for one task while maintaining some minimum success probability for the other tasks. As mentioned above, in calculating the success probabilities needed to evaluate the allocation, the probability of successfully completing one function depends not only upon a sensor's probability of success, but also on the success probabilities of the sensor(s) performing preceding function(s).

The probability calculations can be broken into three steps. First, the success probability for each individual sensor performing a given function must be determined. For each sensor in the first function this is known with some degree of accuracy through analysis of historical data. For the remaining functions the probability determination follows a recursive structure, with the probability of a sensor successfully completing its function calculated using the conditional success probabilities of the sensors assigned to preceding functions and the law of total probability. Second, the overall probability of successfully completing each function is calculated. This involves determining the probability of a set of sensors successfully completing each function using the individual sensor success probabilities calculated in the first step. Finally, the overall task success probability is derived based on the function success probabilities.

The following notation is introduced to formalize the probability calculations. First, let $f \in \{1,\ldots,9\}$ be the set of functions that must be performed in order for a task to succeed. S_i

represents a particular sensor i and C_i^f the set of sensors that perform function (f – 1) and are connected to sensor S_i that performs function f. Finally, $A^f \subseteq C_i^f$ is the subset of connected sensors that accomplished function (f – 1), and $\bar{A}^f \subseteq C_i^f$ the subset of connected sensors that failed to perform function (f – 1), where $A^f \cup \bar{A}^f = C_i^f$ and $A^f \cap \bar{A}^f = \emptyset$.

The probabilities of interest are defined as follows:

- $P_f(S_i)$: the success probability for sensor *i* performing function *f*
- $P_f(S_i|A^f)$: the success probability for sensor *i* performing function *f* given the set of sensors A^f successfully performed function (*f*–1)
- $P_f(A^f)$: the probability that subset A^f successfully performed function (*f*–1)
- $P(f)$: the overall success probability for function *f*
- $P(T)$: the overall success probability for task *T*

For each function beyond the first (whose success probabilities are known for each sensor), the probability of subset A^f occurring (that is, for a particular sensor S_i, the probability that a particular group of connected sensors from the previous function (*f*–1) was successful and the remainder were unsuccessful) is given by

$$P_f(A^f) = \prod_{a \in A^f} P_f(a) \prod_{a \in \bar{A}^f} \left(1 - P_f(a)\right) \tag{3}$$

Because the conditional probabilities $P_f(S_i|A^f)$ are known, sensor probabilities can be determined using the law of total probability as follows:

$$P_f(S_i) = \sum_{A^f \subseteq \mathrm{C_i^f}} P_f(S_i|A^f) P_f(A^f) \tag{4}$$

When each sensor's success probability has been calculated, the overall success probability for each function can be determined. Because of independence, this probability is found using the formula

$$P(f) = 1 - \prod_i \left(1 - P_f(S_i)\right) \tag{5}$$

Finally, with a complete set of function success probabilities, the overall task success probabilities can be calculated following the rules of reliability in a serial system

$$P(T) = \prod_f P(f) \tag{6}$$

The result of this series of equations is a characterization of the feasibility of a resource allocation scheme, and the probability of successfully accomplishing Task 1 under that scheme. Using this scheme as the basis for our determination of a “good” network allocation, the next step is to incorporate it into a methodology that searches for a good allocation. Because the nature of this problem dictates that good solutions be obtained quickly, the use of established optimization techniques are impractical for this problem. Instead, we developed a heuristic approach aimed at quickly finding good solutions, which provides the flexibility needed to operate in this sensitive, dynamic environment.

Heuristic Approach: The heuristic follows a four-stage process that first generates an initial assignment and then works to improve the assignment using a number of different moves. In the application environment, an initial assignment would probably not be generated, but rather the current assignment in use would be used as the initial solution. In stage 1 (see Figure 2), an initial feasible solution is developed. This is done by allocating *assignable* sensors to the secondary tasks (Tasks 2, 3 and 4) using a greedy approach, and random selection of which secondary task is next to be filled, until each task meets its threshold probability, and then allocating the remaining sensors to the primary task (Task 1). This is done in two phases. First, the order in which the secondary tasks are assigned sensors is randomized. Then the first secondary task is assigned sensors randomly until a feasible allocation is achieved. This process is repeated for the second and third secondary tasks. If a secondary task ends up with an infeasible allocation (i.e., the allocation does not yield a success probability for that task above the pre-specified threshold), a new allocation is generated and the process repeated until a feasible solution is found.

In Stages 2 and 3, exchanges are performed in which sensors swap positions in the allocation, relative to Task 1 (see figure 2). By definition, an exchange has to improve the success probability of Task 1 without rendering any secondary task infeasible. The first number used in the description of the exchange indicates the number of sensors assigned to Task 1 (the primary task) that are *removed*; the second indicates the number of sensors that are *added* to the Task 1 allocation. For example, the “1-2 Push” exchange indicates that one sensor assigned to Task 1 is removed from that task (and assigned to another) while two sensors not currently assigned to Task 1 are reallocated from their previously assigned tasks into Task 1. Each time either Stage 2 or Stage 3 is performed, the order in which the exchanges are applied is randomized. This process is repeated until no improvement in the Task 1 success probability is found. If an improved allocation is found, the stage repeats with the first of the ordered exchanges. If no improved allocation is found, the heuristic moves to the next stage.

The search procedure outlined above can be viewed as a variable neighborhood search, where different neighborhoods are used in tandem to reduce the likelihood of becoming prematurely trapped in a local optimum. Hansen and Mladenovic (1998) describe the concept of variable neighborhood local search algorithms, and describe an application to the p-median location problem. The one-sensor exchanges from Stage 2 are useful for rapidly searching relatively small neighborhoods close to the starting solutions. The size of these neighborhoods is $O(n)$, where n is the number of assignable nodes. Stage 3 is used to broaden the search neighborhood, moving further from the starting solution and local optimum. The size of these neighborhoods are larger, $O(n^2)$. Stage 4 continues the search for improvement (returns to stage 2) if there was an improvement in the previous pass. Otherwise, another random starting point is generated and the procedure returns to Stage 1. After a predetermined number of starts, the procedure terminates.

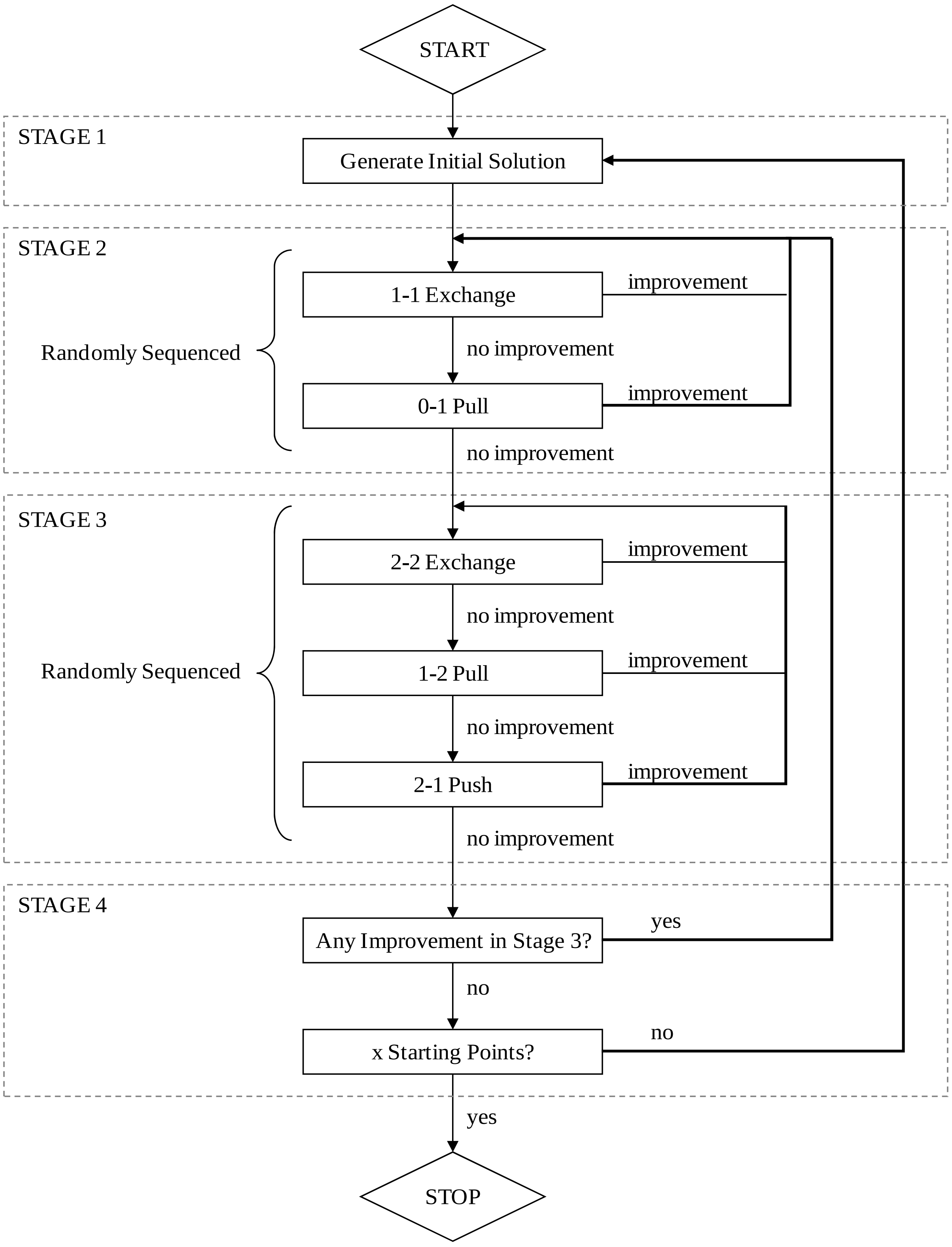

Figure 2. Heuristic Approach

Computational Experiments: Rardin and Uzsoy (2001) provide an excellent tutorial on the computational evaluation of heuristics. Ideally, real-world data (or random variants thereof) should be used to test the performance of a heuristic. However, due to the classified nature of this problem, information concerning sensor probabilities of success, the tasks to which a sensor type may be assigned, the communication connections between sensors, and the size of the network is not available. Hence we use randomly generated test instances that we surmise may reflect the unknown operating environment to a reasonable degree, in the sense that a heuristic that performs well on our test instances should perform well in the actual application. The task requires the identification of key problem parameters and varying those parameters within realistic ranges to create a robust sample of instances against which the heuristic may be tested. Discussions with USSTRATCOM personnel and review of the notional network illustrated in Figure 1 suggest that a real network might include from 12 to 20 assignable sensors that can perform Function 1, with secondary functions generally having fewer available sensors. The ranges of the numbers of sensors assigned to each function in the test instances are given in Table 1, and are notional.

Table 1. Number of Sensors Assigned to Each Function

	Minimum	Maximum
Function 1	12	20
Function 2	8	12
Function 3	12	20
Function 4	12	20
Function 5	4	8
Function 6	6	10
Function 7	4	6
Function 8	2	4
Function 9	4	6

For each test instance, the size of the network, defined by the number of sensors available to perform each function, is generated randomly. The number of sensors available to perform function 1 (the assignable sensors) is fixed at 12, 15 or 20. Using the notional network (Figure 1) as a guide, the average density of the network is estimated to be 50%. We define network density as the proportion of potential network connections that exist. This means that the probability a given sensor in a particular stage can communicate with a sensor in the previous stage is 0.5, as is the probability of it passing information to a particular sensor in the subsequent stage. The network can be made denser or sparser by setting the communication probability at a desired level for generation of the network.

The first function is to detect a potential threat. A particular assignable sensor will have a different success probability of detection depending on the task to which it is assigned. In addition, for sensors performing successor functions success probabilities depend on which sensor(s) (or sensor types) in the preceding function passed the task information to it. The prior probabilities, for the first stage, are generated uniformly over the interval [0.5, 0.9]. For subsequent stages the range of conditional probabilities is over the interval [0.6, 0.8]. Again, these ranges were estimated based on educated best guesses and are notional.

In evaluating the heuristic's performance, several different methods are used. For problems with few assignable sensors, the heuristic solution is compared to an optimal solution obtained by complete enumeration. For larger problems, where explicit enumeration of solutions is not practical, a statistical estimation technique is implemented to derive a probabilistic upper bound to the optimal solution value, and the value of the heuristic solution is compared to the statistical upper bound.

Enumeration of all possible allocation schemes with respect to Task 1 is fairly straightforward, although time consuming. Sensor assignments require enumeration of 4^n possible solutions, with each assignable sensor potentially allocated to exactly one of four tasks, where n is the number of assignable sensors. This process includes feasibility calculations as well as task 1 calculations, so the number of calculations necessary to fully evaluate the possible solutions can be quite large.

Because explicit enumeration is impractical for larger instances, an alternative is to estimate statistically the optimal solution to the large instances. Several approaches have been proposed in the literature to estimate the *minimum* value of a population and generate a confidence interval for that value. The three most common of these assume a continuous probability distribution for the objective function values of the solutions to the problem, differing in the basis for selection of the specific continuous probability distribution. The simplest approach, the *truncation-point* approach, makes no assumptions about the functional form of the probability distribution except that it is continuous and has a lower truncation point at θ (e.g., Robson and Whitlock 1964; Dannenbring 1977). The *extreme-value-theory* approach relies upon a classic statistical result by

Fisher and Tippett (1928) that provides a limiting distribution for the minima of *n* samples of size *m*, as *m* approaches infinity. The third approach, the *limiting-distribution* approach (e.g., Boender et al. 1982), selects a probability distribution that arguably should be an appropriate limiting model as the number of heuristic runs increases towards infinity. These methods estimate the distribution of the solution values by generating multiple solutions to a test instance (i.e., running a randomized heuristic multiple times) and using order statistics derived from these sampled solutions to estimate the parameters of the probability distribution of interest, depending on the particular estimation approach taken. Comparisons of these techniques by Derigs (1985) and later by Ovacik, et al. (2000) and Wilson, et al. (2003) found that an approach presented by Golden and Alt (1979) performed well, and their approach is used here.

The Golden and Alt (1979) approach uses the Fisher-Tippett result to postulate that the distribution of the minima of a number of independent samples from the solution will follow a Weibull distribution. Although several methods have been used in the literature to estimate the Weibull parameters, a very simple approach has proven quite effective. After generating a sufficient number of solutions, those solutions are partitioned into *n* groups each of size *m*. The *n* group minima z_i are found and ordered, such that

$$z_{[1]} \le z_{[2]} \le \cdots \le z_{[n]}$$

The location parameter *a* of the Weibull distribution is then estimated as

$$\hat{a} = \frac{z_{[1]}z_{[n]} - \left(z_{[2]}\right)^2}{z_{[1]} + z_{[n]} - 2z_{[2]}} \tag{7}$$

and the Weibull scale parameter *b* as

$$\hat{b} = z_{[\lfloor 0.63n+1 \rfloor]} - \hat{a} \tag{8}$$

Finally, a lower confidence limit on the optimal solution is given by

$$z_l = \hat{a} - \hat{b} \tag{9}$$

Using this technique, a statistical *lower bound* on the optimal solution is obtained. This technique is easily adapted to create *upper confidence limits* on maximization problems, which is the approach we followed.

In collecting the data required to conduct this statistical estimation process for a given test instance, we generated 250 random initial solutions, each of which was pursued to a local optimum using the 1-1 Exchange only (see Figure 2). Once the 250 local optima were found, they were combined into groups of 10 and the maximum Task 1 success probability within each of the 25 groups became the z_i. These z_i were in turn ordered such that

$$z_{[1]} \geq z_{[2]} \geq \cdots \geq z_{[n]}$$

and the resulting ordered values were used to estimate the Weibull parameters and construct the confidence limits for the optimal solutions as described above. The calculated confidence limits produce a $100(1\text{-}e^{-n})$% confidence interval, so for $n = 25$, the upper confidence limit can be used as an upper bound for the optimal solution. While previous researchers have found a few cases where the true optimum solution lies outside the confidence interval, this was not the case in any of the instances in this experiment where enumeration was possible as a check.

Results: In testing the performance of the heuristic, 50 unique instances each of size 12, 15 and 20 (number of assignable sensors) were run against the criteria previously described. The answers to two questions were sought. First, does the heuristic converge to the known optimal solution for smaller problems (sizes 12 and 15)? Second, does the heuristic converge to the statistical upper bound for larger problems (size 20)?

Initially the 50 test instances of each size were run using 250 initial solutions (restarts). For instances with 12 or 15 assignable sensors, the number of restarts required before the known optimal solution was obtained was measured. In both cases, the number of required restarts was quite low. For size 12 instances, over 80% of the instances found the optimal solution in five or fewer restarts. For size 15 instances, 70% of the instances found the optimal solution in 5 or fewer restarts. The maximum number of restarts needed to find the optimal solution was 33 for the size 12 instances and 39 for the size 15 instances.

The time required to enumerate the size 20 instances was long; none of the instances was completed after two days of computation. Hence the criterion for these problems was the best solution found by the heuristic. Using this measure, over half of the instances converged in five or fewer restarts (see Figure 3). As expected, the heuristic tends to converge to the best solution more slowly for larger instances than for smaller ones. Closer analysis of larger (size 20) problem instances, where more restarts were needed to find the “best” solution, showed that the average variation between the best solution found in 250 restarts and the best solution found in five restarts was less than 0.5%. In other words, the heuristic consistently finds "very good" solutions to the larger problems quickly.

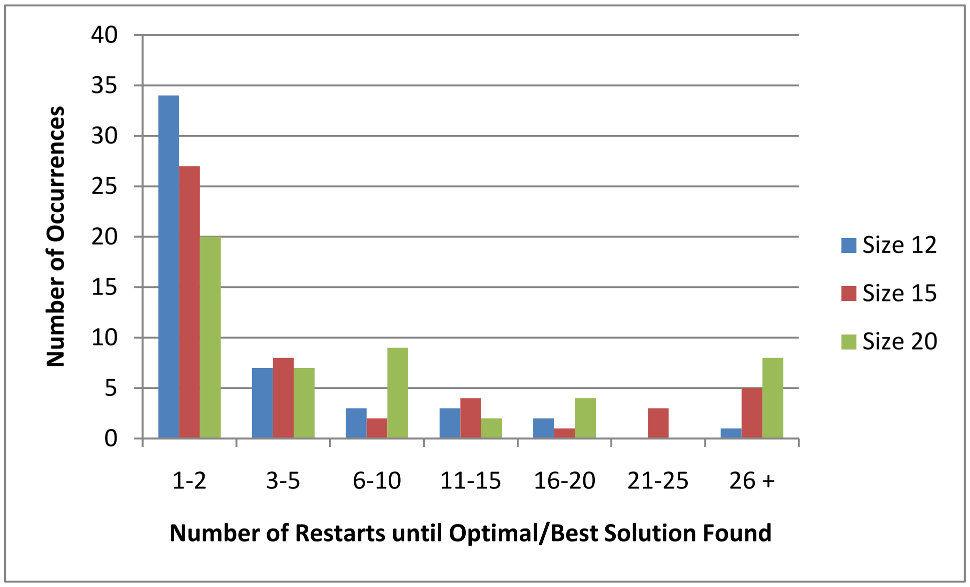

Figure 3. Convergence to Optimal/Best Solution

Using Golden and Alt's method, statistical upper bounds for each of the 150 test instances were estimated, and the heuristic solutions compared to those upper bounds. Figures 4 through 6 compare the statistical bounds and the optimal (or best found) solution obtained for the ten problem instances in which the variation between the statistical upper bound and the best solution were the greatest. Across the 50 instances of size 12, the maximum deviation between the known optimal solution and the statistical bound was 2.3%, but the average deviation across all 50 instances was only 0.24%, including 37 instances where the statistical upper bound and the optimal solution were equal. For the 50 size 15 instances, the maximum deviation between the known optimum and the bound was 1.7%. For the 50 size 20 problems the maximum deviation of the heuristic solutions from the bound was less than 0.9%, significantly better than the worst case in the smaller problems. These results indicate that the bounding technique is valid for generating an estimate of an instance's upper bound, and the improvement in the "worst case" result further indicates that the technique performs well, if not better, as problem sizes increase. Furthermore, even if the best solution found by the heuristic is not the true optimal solution, the difference is so small (typically in the third or fourth significant digit) that the effort necessary to obtain the true optimal is disproportionate to the value gained.

The primary conclusion from these observations is that the heuristic tends to do better as the instance size increases. Larger problems have larger solution spaces as defined by the number of possible feasible solutions, with the larger possibility of near optimal solutions. Randomizing the initial solutions for each run further diversifies the paths that may be taken to obtaining local maxima within each run. If many runs converge to the same local maximum given the widely varied starting points, that local maximum is quite likely to be the global maximum.

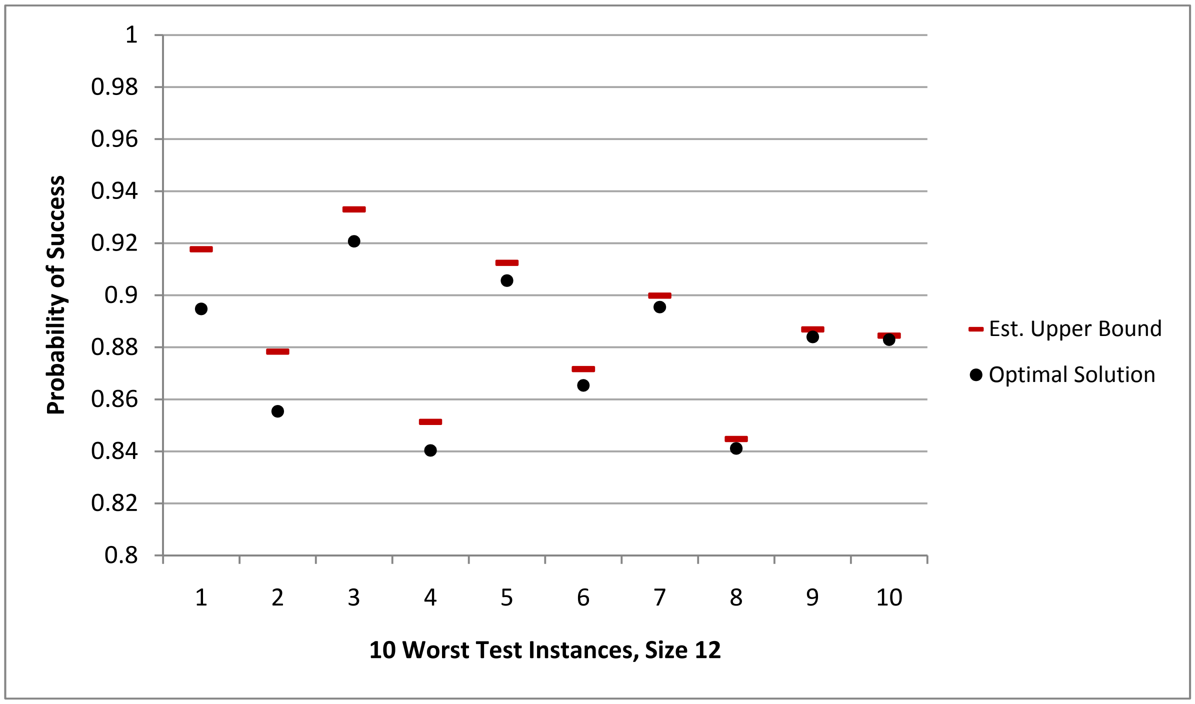

Figure 4: Confidence Limits for the 10 Worst out of 50 Instances, Size 12

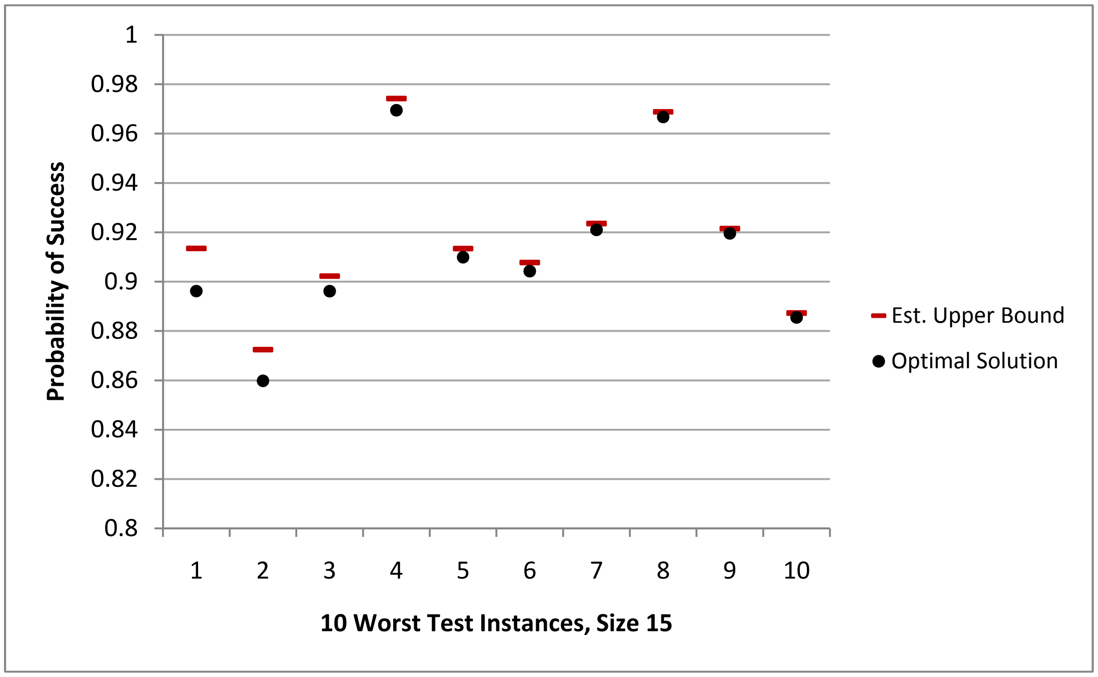

Figure 5: Confidence Limits for the 10 worst out of 50 Instances, Size 15

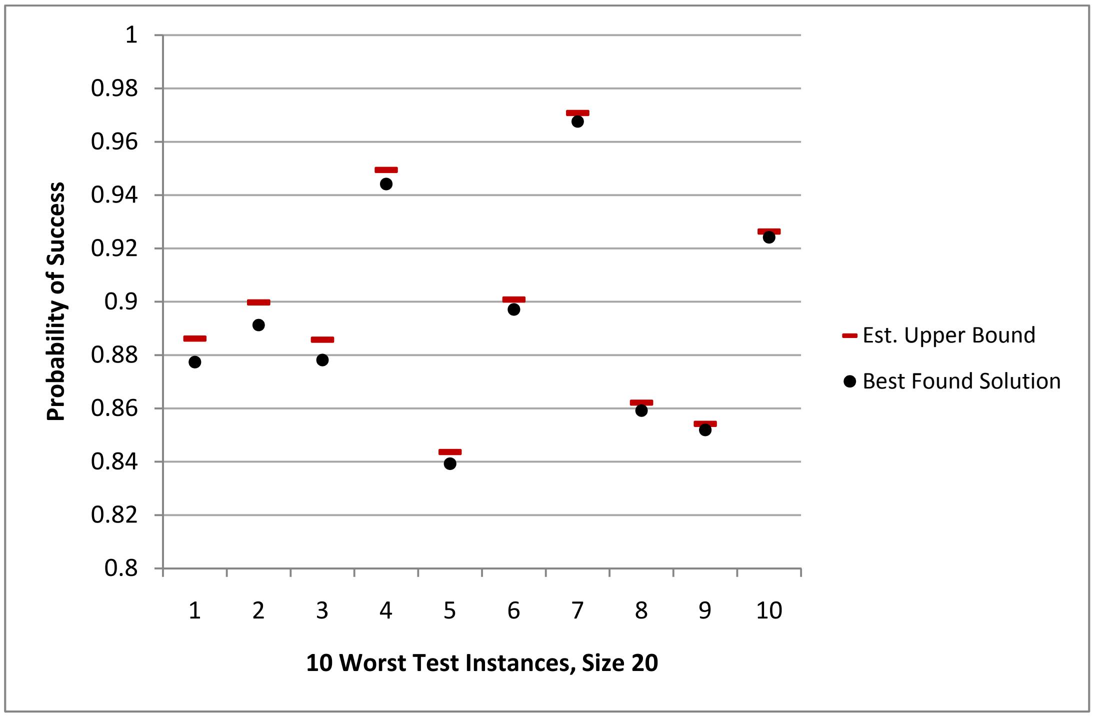

Figure 6: Confidence Limits for the 10 Worst out of 50 Instances, Size 20

Value of the Heuristic: A central aspect of the proposed heuristic is the generation of multiple starting points (feasible solutions). The generation of these initial solutions is done in a greedy manner, randomly assigning sensors to secondary tasks until the probabilities of success for those tasks each meet their thresholds. As a result, it is likely that the number of sensors assigned to each task is at or near the minimum required to achieve that feasible probability. This raises the question of how much improvement the search heuristic yields over a simple random solution. A comparison of random starting solutions was made against the solutions obtained running the heuristic with a common computation time limit imposed on both procedure. Fifty test instances of each size were compared under two scenarios. In the first, the heuristic was run to completion for each test instance (250 restarts each) and the total computation time was recorded, then random solutions were generated for the same period of time. Second, the heuristic was run until the optimal (or best known) solution was found for the first time, and then random solutions were generated across that time interval.

For the size 12 problems, the random solutions included the optimal in 43 of 50 instances. The random solutions generated in the shorter period (second scenario) included the optimal in 17 of 50 instances. For size 15 problems, the random solutions included the optimal in 10 of 50 instances. The random solutions for the second scenario included the optimal in 2 of 50 instances. For the size 20 problems, the random solutions failed to include the best known solution for any instance under either scenario. Hence for the larger instances for which the heuristic is intended there is certainly value in employing the heuristic as opposed to simply generating random solutions in the hopes of quickly discovering the best possible solution.

Computation Times for the Heuristic versus enumeration: For ease of development, all computer programming was performed in Visual Basic (VB). The size 12 problems took from 1 to 2 minutes to enumerate, while the heuristic solved the problems in less than 2 seconds (smallest ratio greater than 30/1). The size 15 problems took from 10 to 90 minutes to enumerate, while the heuristic solved the problems in less than 10 seconds (smallest ratio greater than 60/1). The size 20 problems took over 24 hours to enumerate (none were completely enumerated), while the heuristic solved the problems in less than 2 minutes (smallest ratio greater than 720/1). Since VB is an interpreter, one would expect that the computation times experienced here are roughly an order of magnitude greater than they would be if the procedure were coded in a compiler language such as C++ or Fortran.

Network Changes: The ability to generate a near-optimal allocation in a static environment is valuable, but the capability to react to changes in the network environment is even more important in the application domain for which this heuristic is intended. Not only can the proposed heuristic quickly find a good solution to a particular sensor network, it is flexible enough to quickly find a good solution to a "new" network when the environment changes suddenly.

One environmental change of interest is the addition of a sensor to the "current" network. This may be either an assignable sensor or a shared sensor, where a sensor may be re-activated after undergoing maintenance, or an orbiting sensor enters a particular region of interest. In either case, there is no negative impact on the network's performance, since the success probability for any task will not decrease by adding a sensor. Thus, action is not necessary immediately in terms of reallocating sensors across tasks. If the new sensor is assignable, it can immediately be allocated to Task 1. This allocation can then be used as a starting point for the heuristic to find an even better allocation with respect to the Task 1 success probability.

Another change is the removal of a sensor (e.g., destroyed by attack, orbiting sensor moves out of range). There are three scenarios to be considered: the removal of a sensor assigned Task 1; the removal of a sensor assigned a secondary task; or the removal of a *shared* sensor. If a Task 1 sensor is removed, the allocation remains feasible, but the success probability for Task 1 will decrease. The removal of a secondary task sensor is likely to result in infeasibility, necessitating a reassignment of sensors to restore feasibility. Finally, removal of *shared* sensors may yield both a decrease in the Task 1 success probability and infeasibility.

When the removal of a sensor, whether assignable or shared, results in an infeasible allocation with respect to the secondary tasks, action must be taken to reallocate the remaining sensors so as to find an optimal/near optimal, feasible solution to the new network. Of course, reassessment of the feasibility requirements may be appropriate too, if the new network is such that the previous constraints are too restrictive to allow reasonable probabilities of successful completion of Task 1. **However, that is beyond the scope of this paper**.

Future Work: A natural extension of the current effort is to investigate issues of *objective function modification*. While this paper focused on the development of an approach geared to maximizing the probability of successfully accomplishing one primary task, *objective function modification* may more accurately capture the nature of the problem. The goal of the sensor network may be to maximize the probabilities of success for all four tasks (i.e., maximize the minimum of the four task probabilities). This necessarily assumes that all tasks are of equal importance, and that each has the same minimum threshold requirement. Ultimately, the decision-maker must determine the appropriate characterization of the objective function, and what ***is*** appropriate will change as tactical and strategic situations change.

Conclusions: A heuristic approach that was developed to allocate sensors across competing tasks has been shown to achieve optimal/near optimal solutions, subject to the constraints of the network. Solutions are obtained in a fraction of the time that would be required to guarantee optimality using an integer programming model.

The application of the heuristic to networks representative of the real-world problem has been shown to be effective in determining the best assignment of sensors to tasks. Also, as events occur that impact the performance of the network, it is necessary to have a tool in place that can re-evaluate that network and recommend changes to the sensor assignments. The heuristic proposed here accomplishes that, providing a decision support tool that can reassign sensors to tasks quickly and effectively.

References:

Aarts, E., J.K. Lenstra (Eds.) (2003). Local search in combinatorial optimization, Princeton, NJ, Princeton University Press.

Barr, R.S., B.L. Golden, J.P. Kelly, M.G.C. Resende and W.R. Stewart Jr. (1995). "Designing and reporting on computational experiments with heuristic methods," *Journal of Heuristics*, 1, 9-32.

Boender, C.G.E., A.H.G. Rinnooy Kan, L. Stougie and G.T. Timmer (1982). "A stochastic method for global optimization," *Mathematical Programming*, 22, 125-140.

Dannenbring, D.G., (1977). "An evaluation of flow shop sequencing heuristics," *Management Science*, 23(11), 1174-1182.

Derigs, U. (1985). "Using confidence limits for the global optimum in combinatorial optimization," *Operations Research*, 33(5), 1024-1049.

Dulin, Johnathon L. (2008), *Global Sensor Management: Real-Time Reallocation of Military Assets among Competing Tasks and Functions*. Ph.D. Dissertation, Operations Research Program, N.C. State University, Raleigh, NC.

Fisher, R.A. and L.H.C. Tippett (1928). "Limiting forms of the frequency distribution of the largest or smallest member of a sample," *Proceeding of the Cambridge Philosophical Society*, 24, 180-190.

Golden, B.L. and F.B. Alt (1979). "Interval estimation of a global optimum for large combinatorial problems," *Naval Research Logistics Quarterly*, 26(1), 69-77.

Gordon-Spears, D. and K. Kiriakidis (2004). "Reconfigurable robot teams: modeling and supervisory control," *IEEE Transactions on Control Systems Technology*, 12(5), 763-769.

Hansen, P., N. Mladenovic (1998). "An introduction to variable neighborhood search," Voss, S. et al. (Ed.), Meta-heuristics Advances and Trends in Local Search Paradigms for Optimization, Kluwer, Dordrecht, 433-458.

Howard, A., M.J. Mataric and G.S. Sukhatme (2002). "An incremental deployment algorithm for mobile robot teams," *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2849-2854.

Izquierdo-Fuente, A. and J.R. Casar-Corredera. "Optimal Radar Pulse Scheduling Using a Neural Network," *Proceedings of the 1994 IEEE Conference on Neural Networks*, 4588-4591.

Joint Publication 3-01 (2007). *Countering Air and Missile Threats*, Headquarters, Department of Defense, Washington, DC.

Joint Publication 3-14 (2002). *Joint Doctrine for Space Operations*, Headquarters, Department of Defense, Washington, DC.

Kapur, C.K. and L.R. Lamberson, *Reliability in Engineering Design* (1977). John Wiley and Sons, New York, NY.

Miranda, S.L.C., C.J. Baker, K. Woodbridge and H.D. Griffiths (2007). "Comparison of scheduling algorithms for multifunction radar," *IET Radar, Sonar and Navigation*, 1(6), 414-424.

Orman, A.J., A.K. Shahani and A.R. Moore (1998). "Modelling for the control of a complex radar system," *Computers and Operations Research*, 25(3), 239-249.

Ovacik, I.M., S. Rajagopalan and R. Uzsoy (2000). "Integrating interval estimates of global optima and local search methods for combinatorial optimization problems," *Journal of Heuristics*, 6, 481-500.

Pinedo, M. (2002). Scheduling: theory, algorithms and systems. Upper Saddel River, NJ: Prentice Hall.

Rardin, R.L. and R. Uzsoy (2001). "Experimental evaluation of heuristic optimization algorithms: a tutorial," *Journal of Heuristics*, 7(3), 261-304.

Robson, D.S. and J.H. Whitlock (1964). "Estimation of a truncation point," *Biometrika*, 51, 33-39.

Wilson, A.D., R.E. King and J.R. Wilson (2003). "Case study on statistically estimating minimum makespan for flow line scheduling problems," *European Journal of Operational Research*, 155, 439-454.

Wolsey, L.A. (1998). Integer Programming, New York, NY: John Wiley & Sons, Inc.

Zweben, M., E. Davis, E. Daun and M.J. Deale (1993). "Scheduling and rescheduling with iterative repair," *IEEE Transaction on System, Man, and Cybernetics*, 23(6), 1588-1596.